MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2019/2020

ECE1026 ALGORITHMS AND DATA STRUCTURES

(All Sections / Groups)

9 MARCH 2020
9:00 A.M – 11:00 A.M.
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This question paper consists of 9 printed pages including the cover page with 4 questions only.

2. Attempt all FOUR (4) questions. All questions carry equal marks and the distribution of marks is given in the questions.

3. Please print all your answers in the answer booklet provided.

4. State all assumptions clearly.

Question 1

(a) A Finite State Automaton (FSA) is shown in Figure Q1(a).

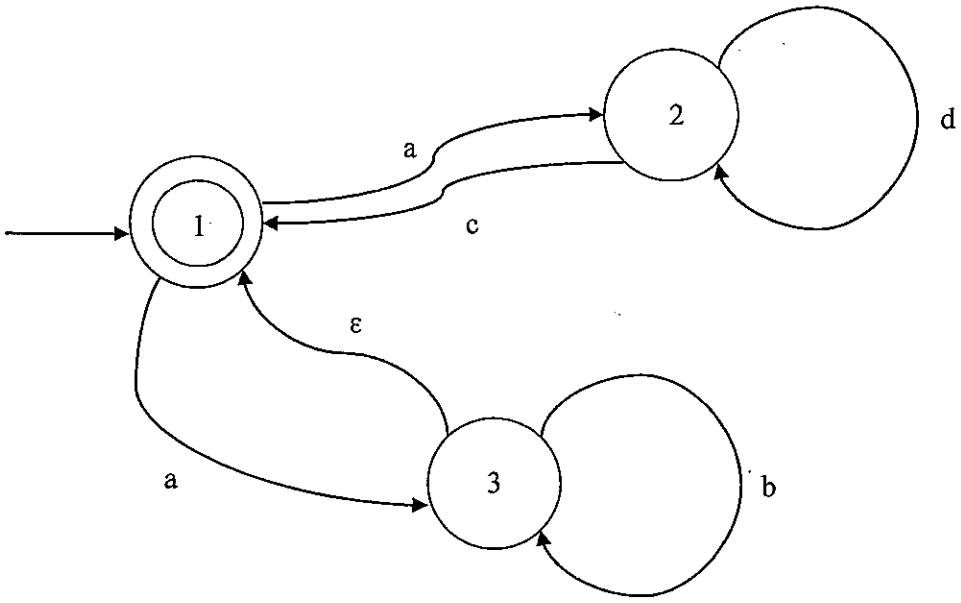


Figure Q1(a)

Based on Figure Q1(a):

(i) Write the Regular Expression (RE) for this FSA.

[7 marks]

(ii) List the set of all states (Q), the set of all input symbols ($\sum$), the initial state (q_0), and the set of final states (F) for this FSA.

[5 marks]

(iii) Decide whether this is a Deterministic FSA (DFSA) or Non-deterministic FSA (NFSA) by providing proofs that could be observed in this FSA.

[4 marks]

(b) Draw a NFSA that will accept the RE = (1+3)*.

[4 marks]

Continued...

(c) Given the FSA Controller of a Turing Machine (TM) in Figure Q1(c), write down the content of the tape and the position of the read/write head at the end of the execution when the machine stops. Assume that the read/write head is pointing to the first left most non-blank square.

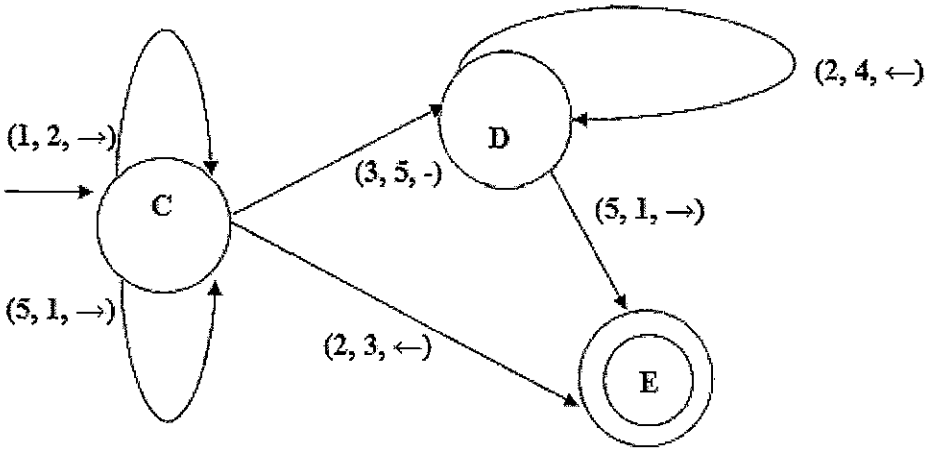
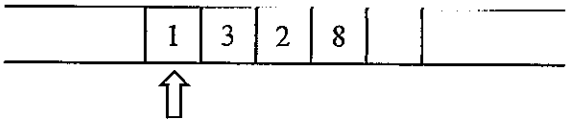


Figure Q1(c)

Initial content of the tape:



[5 marks]

Continued...

Question 2

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//Answer for part (a)(i)

struct Book *addBook(char title[], float price, int stock);
void inventory(struct Book *);

int main(){
   struct Book *head=NULL;
   struct Book *current=NULL;
   current = head = addBook("",0.0, 0);
   current->next = addBook("The Silent Patient", 17.90, 18);
   current = current->next;

   current->next = addBook("Daisy Jones & The Six", 23.50, 7);
   current = current->next;

   current->next = addBook("The Testaments", 49.90, 25);
   current = current->next;

   //Answer for part (a)(iii)
   //A while loop to determine the new book prices at 25% discount

   inventory(head);

   return 0;
}

struct Book *addBook(char title[], float price, int stock){
   //Answer for part (a)(ii)
}

void inventory(struct Book *current){
   //Answer for part (a)(iv)
}
```

Program Listing Q2(a)

Table Q2(a)

Title	Price	Stock
The Silent Patient	17.90	18
Daisy Jones & The Six	23.50	7
The Testaments	49.90	25

Continued...

(a) Answer questions (i) to (iv) below with reference to Program Listing Q2(a).

(i) Based on Table Q2(a), write the C code to define a structure **Book** in order to keep track of the available books in a bookstore in the form of a linked list.

[5 marks]

(ii) Complete the function definition for the function **addBook()** that takes in the string **title**, float **price** and integer **stock**, creates and initializes a new node with these values and then returns the pointer to the new node to the **main()** function.

[5 marks]

(iii) The owner would like to give a **25%** discount on all titles. Write the C code that reset the current pointer back to the start of the linked list in the **main()** function. Then, using a **while** loop, print out the discounted prices of the books in the basic linked list created in Program Listing Q2(a) on the monitor. A sample output is as shown in Figure Q2(a)(iii).

[4 marks]

```
Discounted price for the book entitled The Silent Patient is
RM13.42.
Discounted price for the book entitled Daisy Jones & The Six is
RM17.62.
```

Figure Q2(a)(iii)

(iv) Complete the function definition for the function **inventory()** that check and remove the books which are out-of-stock from the linked list.

[4 marks]

(b) Study the code snippet in Program Listing Q2(b).

```
typedef unsigned int Set;
#define SETSIZE 32

void printSet(Set s)
{
    int j;
    printf("{ ");
    for(j=0; j<SETSIZE; j++)
        if((1<<j)&s) printf("%d ",j);
    printf("}\n");
}
```

Program Listing Q2(b)

Continued...

Using linear bitmap implementation of sets and 32-bit data byte, write C statements to complete the following tasks. A sample output is as shown in Figure Q2(b).

```
A = { 5 11 30 }
B = { 3 5 7 10 30 }
A union B = { 3 5 7 10 11 30 }
A intersection B = { 5 30 }
B = { 3 4 5 7 10 30 }
```

Figure Q2(b)

(i) Declare and initialize set **A** = {5, 11, 30}, **B** = {3, 5, 7, 10, 30} and print them.
[4 marks]

(ii) Find and print the union of **A** and **B**.
[1 mark]

(iii) Find and print the intersection of **A** and **B**.
[1 mark]

(iv) Insert element 4 into set **B** and print set **B**.
[1 mark]

Continued...

Question 3

(a) The GCD (Greatest Common Divisor) of two numbers is the largest positive number that divides both the numbers without leaving any remainder. It can be expressed mathematically as shown in the equation below:

$$gcd(a,b) = \begin{cases} b & a = 0 \\ a & b = 0 \\ a & a = b \\ gcd(a-b,b) & a > b \\ gcd(a,b-a) & a < b \end{cases}$$

(i) Construct a direct *recursive* $gcd(a,b)$ function in C programming language based on the equation above.

[6 marks]

(ii) Write a C statement to call the *gcd* function you wrote in part (i). Prompt the user for the values of $\boldsymbol{a}$ and $\boldsymbol{b}$, and display the result on the screen.

[2 marks]

(iii) Determine the value of $\boldsymbol{gcd(98,56)}$ by showing step by step function calls until it reaches the final condition (base case).

[4 marks]

(iv) Is it possible to rewrite the *gcd* function without recursion? Justify your answer.

[2 marks]

(b) In your own words, describe backtracking algorithm and its area of application.

[4 marks]

(c) Consider the maze in Figure Q3(c). '**X**'s are walls, blanks are passages, '**S**' is the start point and '**E**' is the end point. *Sketch* the solution path (including the backtrack path) from point '**S**' to point '**E**' using a *backtracking* algorithm with the following search order: 1) West (left), 2) North (up), 3) East (right), 4) South (down).

[7 marks]

X	X	X	X	X	X	X	X
X			X				X
X			X	X	X		X
X	X						X
X	X		X	X			X
X	X		X	X			X
X	S		X	X	E		X
X	X	X	X	X	X	X	X

Figure Q3(c)

Continued...

Question 4

(a) Suggest a suitable *search* algorithm for each of the arrays: *Data A* and *Data B*. Explain your answer.

[4 marks]

$$Data\ A = [299 \quad 77 \quad 10 \quad 14 \quad 2222 \quad 384 \quad 25]$$

$$Data\ B = [0.6 \quad 6 \quad 6.6 \quad 16 \quad 60 \quad 666 \quad 6000]$$

(b) Consider the following incomplete C program as shown in Figure Q4(b).

```
#include <stdio.h>
#include <string.h>
#define SIZE 5

struct Furniture {
   char name[50];
   int quantity;
   float price;};

//Function definition for stockSearch - Q4(b)(i)
//Function definition for display - Q4(b)(ii)

int main() {
   char search_furniture[30];
   int result;
   struct Furniture stock[SIZE] = {
      {"Chair", 50, 25.00},
      {"Dining Table", 10, 2999.99},
      {"Divan Bed", 30, 3555.55},
      {"Sofa", 15, 5666.00},
      {"Divan Bed", 25, 7888.88}};

   printf("Enter furniture: ");
   gets(search_furniture);

   result = stockSearch(search_furniture, stock);
   display(result, stock);

   return 0;}
```

Figure Q4(b)

(i) Write the C function **`stockSearch`** that performs *binary search* to search for the furniture in the array of stock. The function takes in the **`search_furniture`** and array of **`Furniture`** structures. If a match is found, the function returns the index of the matching element, otherwise it returns **`-1`**. Use *iterative* method with a **`while`** loop.

[7 marks]

(ii) Write the C function **`display`** that prints out the details (i.e. furniture name, quantity, price) of the matching element. The function takes in the **`result`** (the index of matching element) and array of **`Furniture`** structures. If there are no results, display "**`Not Found`**" instead.

[5 marks]

Continued...

(c) Sort by hand the word "**COMPUTERS**" in *ascending* order using the following sorting algorithms. Show all steps involved.

(i) Selection sort [3 marks]

(ii) Insertion sort [3 marks]

(iii) Quick sort (the middle element as pivot) [3 marks]

End of paper